Using Pre-Built Program Templates to Teach Numerical Methods

David L. Silverstein
University of Kentucky

Abstract

Most engineering programs require students to learn some form of structured programming early in their curriculum, but in many cases, students do not use their newly acquired programming skills again. While outside of computer-related majors there may not be a need to maintain programming skills, programming is still an effective way to make certain that students understand how a numerical method is arriving at a solution. A method called "template-based programming" was developed to enable use of high-level computer languages in courses where programming is not explicitly part of the course objectives. In this method, a student is given a fully functioning program, or template, that only lacks the functional code for a numerical method to solve a particular type of problem. Since the work of developing the interface and other portions of the program has been completed for the student, all they need to concentrate on are the aspects of the programming project that contribute toward the course objectives. Examples of how this approach has been used in numerous chemical engineering courses will be presented, including templates developed in Compaq Visual FORTRAN, Microsoft Visual Studio.NET, and Microsoft Excel.

Introduction

In the University of Kentucky chemical engineering curriculum, students are required to take a course in computer programming prior to taking their first "core" chemical engineering course. Subsequent to that course, it is possible that a student will never to be required to write a complete program from "scratch" again. This makes some sense in chemical engineering and other disciplines where greater than 90% of practitioners never program in a high-level language.[1]

Computer programming concepts in some form are still required according to Accreditation Board for Engineering and Technology (ABET) criteria.[2] While most curricula still require high-level languages such as C, C++, and Visual Basic,[3] an increasing number of programs are choosing to teach scripting languages within mathematics applications such as Maple, MATLAB, and Mathematica.[4]

Even though most practicing engineers will not program, it is often argued that programming is an effective means of teaching problem formulation and problem

solving.[5] Programming languages are "a novel formal medium for expressing ideas about methodology."[6] Other researchers have presented counterarguments to that premise.[7] Nonetheless, the ability to develop a computer application is not a pre-requisite for most jobs, and typically is not required to meet course objectives outside of the programming course.

Since programming is still taught, students have a valuable skill that can be utilized to achieve course objectives. A course involving numerical methods is a good fit for programming assignments, but asking students to develop a computer application requires substantial time commitment to activities outside course objectives. Students enrolled in a thermodynamics course should spend their time pondering the meaning of entropy instead of manipulating FORMAT statements. This paper describes the most recent extensions of an approach demonstrated to allow focus on course objectives while using computer programming, "Template-Based Programming".[8,9]

Template-based programming refers to the practice of providing students a completely functioning application that compiles and displays an operational dialogue window immediately upon retrieval from a class web site or other source. The only element lacking from the program is the code from whatever routine is designated to perform the required calculations. Typically, this is code for a numerical method acting upon inputs provided by the user in the dialogue box. This differs from modifying existing programs or scripts in that students still start from a blank file (except for comments) and generate all required logical steps themselves instead of altering existing code. Clever students can alter existing code without understanding the underlying logic in an algorithm. Students are responsible for all elements requiring assessment under the course and project objectives.

Previous publications[8,9] have described the use of templates in Compaq Visual FORTRAN and Microsoft Visual Basic. This paper describes the development issues associated with those templates as well as new ones developed using Visual Basic for Applications running under Microsoft Excel.

Use of Templates

At the time of writing, templates have been used in courses in stoichiometry and process modeling. Both have course objectives associated with numerical methods.

The process modeling course has used two templates. The first is an exercise in establishing a value for the machine epsilon, the smallest value that can be distinguished from 1 by a computer. This serves as an introduction to the process of using these templates as well as emphasizing a key point about the limits of computer-based calculations. This template has been developed in both FORTRAN and Visual Basic in order to be available to students having taken courses in either language. Figure 1 shows the dialogue included in both the FORTRAN and Visual Basic Templates. Figure 2

shows the contents of the module students are instructed to "fill in" to complete the program.

The second template requires the student to implement Gauss-Siedel iteration for a system of linear equations. To date, this has only been implemented using FORTRAN, as shown in Figure 3. A version of the template functioning in Visual Basic for Applications within Excel is currently under development. The programming assignment reveals to students how well they know (or do not know) how to use nested DO loops.

In the stoichiometry class, students are expected to use a root-finding technique to solve a cubic equation of state. Three versions of this template have been used, two of which were described previously. Both the Visual Basic and FORTRAN versions of the template require the students to use Newton's method to solve for an unknown amongst temperature, pressure, and specific volume using the Soave-Redlich-Kwong equation of state. The dialogue generated is shown in Figure 4. The most recent version of the template combines a Microsoft Excel Spreadsheet template to calculate parameters for the Peng-Robinson equations of state; the student then works with a Visual Basic for Applications (VBA) template to solve for compressibility using Newton's Method. Figure 5 shows a portion of the Excel spreadsheet including a "button" which activates the VBA routine, shown in Figure 6.

Developing Templates

The choice of what environment to use to develop templates should be dependent on the language in which students have prior instruction and experience. Since the point of this approach is to minimize time spent away from course objectives, it would be counter-productive to teach extensively about a new language or environment. Additionally, there are challenges associated with each language which should be considered by the person preparing the template.

Compaq Visual FORTRAN is the most challenging environment amongst these three in which to prepare a template. This environment, more akin to Visual C/C++ 6.0 and earlier versions than the current .NET environment, requires understanding of how to develop and interface with non-trivial user dialogues. It would be easier for most non-professional programmers to attempt to modify existing programs to suit their needs, though even this approach requires a substantial time investment. While FORTRAN is an excellent language in which to write numerical algorithms, this particular implementation of the language is a non-ideal platform for application development. More recent implementations by Lahey make this portion of template development much easier for the visual programmer.[10] Other issues that arise are project source file paths which may not automatically update when files are transferred from the developer's computer to the student's; incompatibilities between versions of the compiler (in particular, improved error checking in later versions); and compatibility issues on lab computers when more current versions of other application development environments are installed.

Visual Basic has from its beginning been a relatively easy way of developing applications which function well under Microsoft Windows. Templates are also easy to develop under VB, though there is one issue that often arises. Issues of variable scope are often encountered since the template approach recommends that the student's work be self-contained in its own module. These issues are most directly addressed by making important variables global, abandoning any pretense of following best practices in object oriented application development.

Both of these environments suffer from one flaw. While the program developed is useful to the student and does facilitate achievement of project objectives, it passes up the opportunity to develop a skill that would actually be used in a workplace environment. Students can solve very interesting and complex problems using Maple, Mathcad, Matlab, or Mathematica. They can even develop sophisticated applications using an application development environment of some sort. When they enter most plant situations, however, they will likely not have these resources available to them. They will, in most cases, have access to Microsoft Excel, which has VBA available to them. Teaching students to implement their numerical methods as an extension to Excel not only maintains the focus on course objectives, but enables them to perform similar functions later in any work environment in which they may find themselves.

The most recent template-based project combined elements of spreadsheet "programming", taught in an introductory class, as well as VBA programming, building off concepts taught in a programming course. Several types of programs are possible using VBA under Excel, including macro automation, custom functions, and custom subroutines. The latter two are of interest for numerical methods. The function approach enables access to a numerical routine which behaves in the same manner as an intrinsic function such as SIN(). It takes inputs from the values or references in the argument list and returns a single value (or an error message). A function cannot alter any cell in the spreadsheet other than the cell it contains. A subroutine can alter any property of the spreadsheet, including values, formulae, and formats. For iterative calculations, it will often make more sense to use a subroutine for a program to remain general purpose, while a function is an elegant approach to many common solution methods.

Summary

The use of programming templates to facilitate use of high-level languages to teach numerical methods is presented. Templates enable focus on project and course objectives while still taking advantage of the pedagogical benefits of programming. The currently preferred platform is Visual Basic for Applications under Microsoft Excel, building on student instruction in both Excel and Visual Basic. Students demonstrate algorithmic understanding by programming, and build their skills with a software tool with near-universal availability. Other environments, including Compaq Visual Fortran and Visual Basic have programming issues which can be overcome with some programming skill and investment of developer time. The template product, regardless of the development

environment, is useful for enabling focused instruction in numerical methods. All of the templates described in this paper are available on the author's web site.[11]

References

[1] Davis, J., Blau, G., & Reklatis, G.V. Computers in undergraduate chemical engineering education: A perspective on training and applications. Technical report, CACHE Corporation. Draft 3.1. (1993).

[2] *Criteria for Accrediting Engineering Programs*, Accreditation Board for Engineering and Technology, Inc., Baltimore, MD (2002). URL: http://www.abet.org/

[3] URL: http://www.che.utexas.edu/cache/survey.html; CACHE: Survey Results

[4] Dahm, K.D., Hesketh, R.P., and Savelski, M.J., *Chem. Eng. Ed.*, **36**, p. 192 (2002).

5 Stephanopoulos, G. & Han, C. Languages and Programming Paradigms. In B. Carnahan (Ed.), *Computers in Chemical Engineering Education*, Austin, Texas: CACHE Corp. (1996).

[6] Abelson, H. and Sussman, G., Structure and Interpretation of Computer Programs, Cambridge, MA, MIT press (1985).

[7] Urban-Lurain, M. and Weinshank, D.J., "Do Non-Computer Science Students Need to Program?", *J. Eng. Ed.*, **88**, p. 535 (2001)

[8] Silverstein, D.L. "Template Based Programming in Chemical Engineering Courses". Proceedings of the *2001 ASEE Annual Conference & Exposition*. American Society for Engineering Education, (2001).

[9] Silverstein, David L., "Increasing Time Spend on Course Objectives by Using Computer Programming to Teach Numerical Methods", *Chem Eng Ed*., **37**, p. 214 (2003)

[10] URL: http://www.lahey.com/; Lahey/Fujitsu Fortran v7.0

[11] URL: http://www.engr.uky.edu/~silverdl/TBP

DAVID L. SILVERSTEIN
David L. Silverstein is currently an Assistant Professor of Chemical and Materials Engineering at the University of Kentucky College of Engineering Extended Campus Programs in Paducah. He received his B.S.Ch.E. from the University of Alabama in Tuscaloosa, Alabama; his M.S. and Ph.D in Chemical Engineering from Vanderbilt University in Nashville, Tennessee; and has been a registered P.E. since 2002. He has over twenty years experience in microcomputer programming, most recently in development of a prototype automatic custom videotape editing and production device. In addition to teaching and research in interfacial phenomena, Dr. Silverstein is developing a computer framework for applying learning styles to a multimedia computer-based supplement to engineering courses.

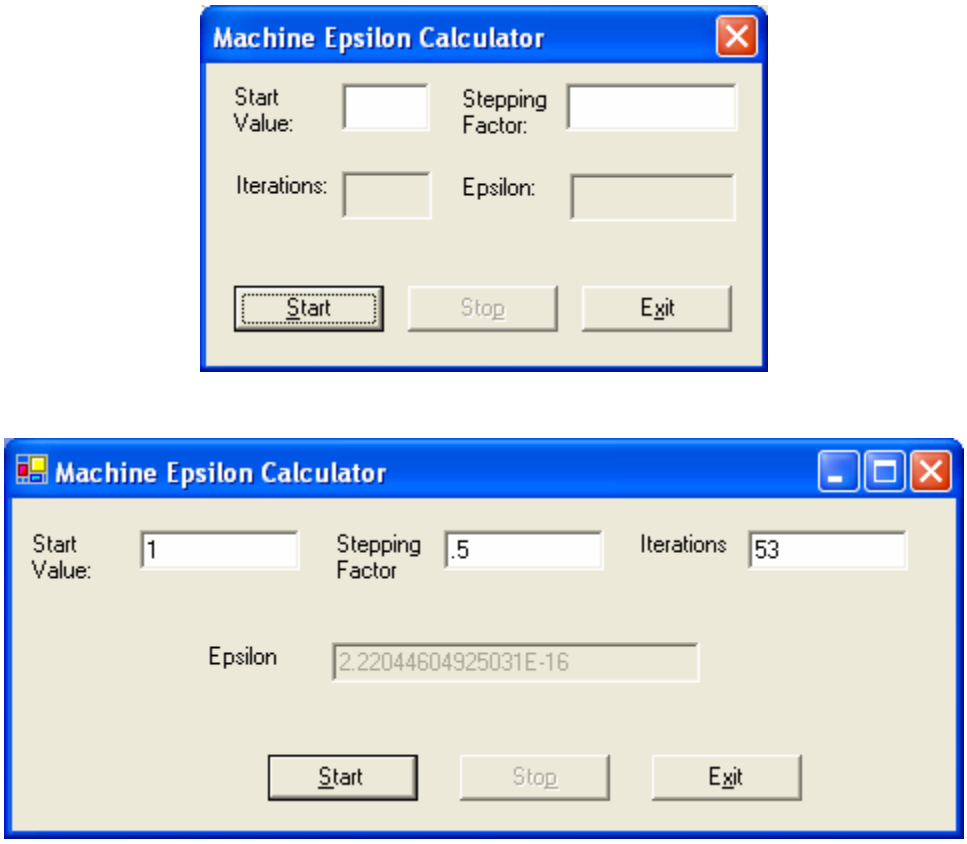

Figure 1. Dialogue boxes for machine epsilon calculator templates in FORTRAN (above) and Visual Basic (below). Some differences arise from development choices to simplify formatting of output data.

```
Public Module EpsilonCode
    'Routine to actually run the epsilon determination calculations
    '
    'It accepts its start value and stepping factor from the dialog
    'It must return the value of epsilon and the number of iterations
    'Minimal error checking is performed

    Public Sub EpsilonCalc(ByVal StartValue As Double, ByVal
SteppingFactor As Double, ByRef Epsilon As Double, ByRef Iterations As
Double)
     'The following variables initialized from the form are available
to you.
     ' StartValue = a double precision real value
     ' SteppingFactor =  a double precision real value between 0 and 1
     '!You must specify values for the following two variables before
     ' exiting the subroutine so the form can be updated
     'Epsilon = a double precision value indicating the machine epsilon
     'Iterations=an integer indicating the number of times your
calculation looped before finding the answer
     'For an added challenge, make the stop button allow you to exit
the subroutine!

        'YOUR CODE STARTS HERE



        'YOUR CODE ENDS HERE
    End Sub
End Module
```

Figure 2. Visual Basic module which students complete to implement the numerical method. The solution for this calculation is about 6 lines of code. Note that the comments inform the student regarded the variables used to access dialogue box variables.

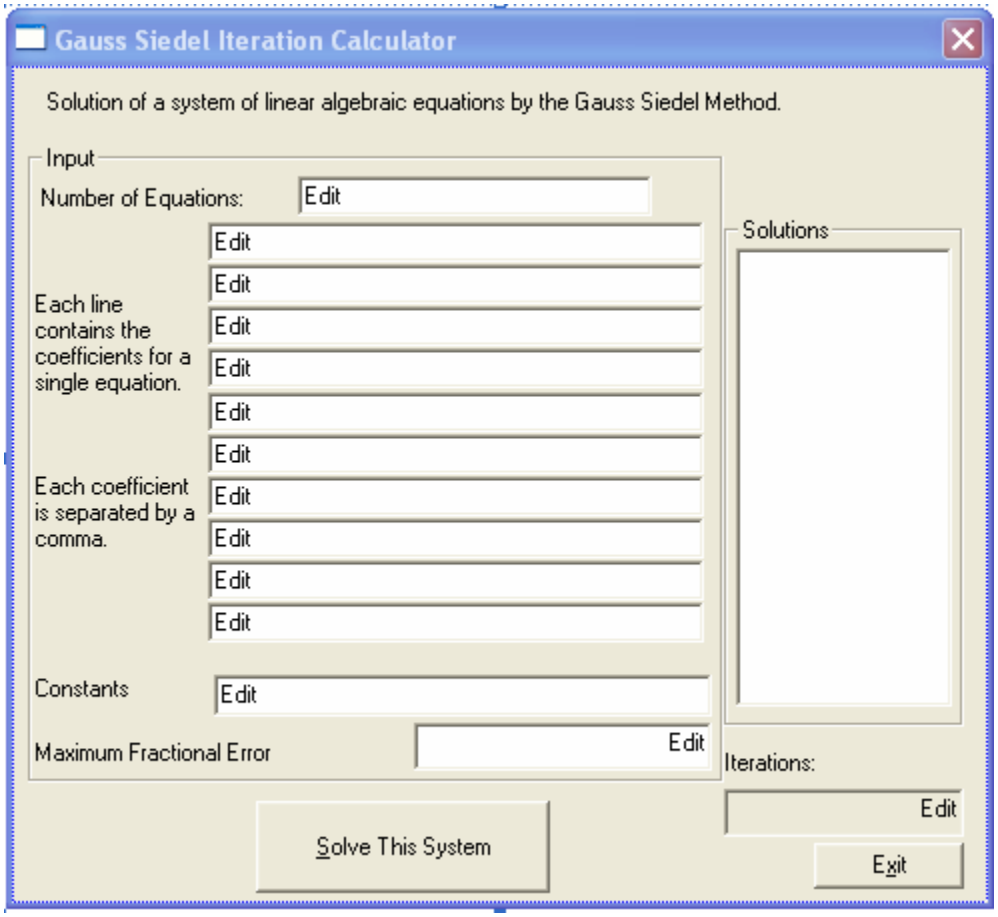

Figure 3. Dialogue for solution of a system of linear equations under Compaq Visual FORTRAN. The data entry method is somewhat awkward and could be improved by using a grid interface such as a spreadsheet (coming February 2004).

Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition

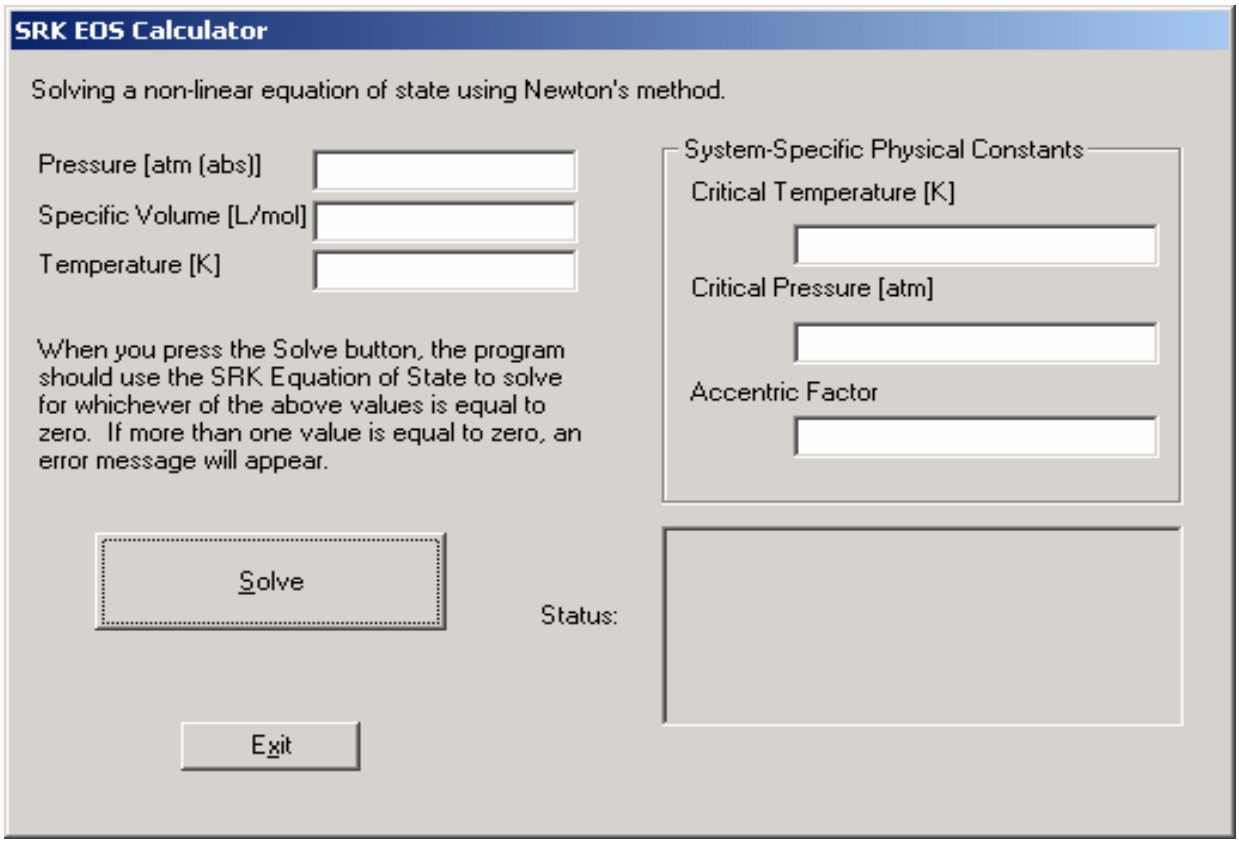

Figure 4. Dialogue for SRK calculator under Visual Basic.

Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition

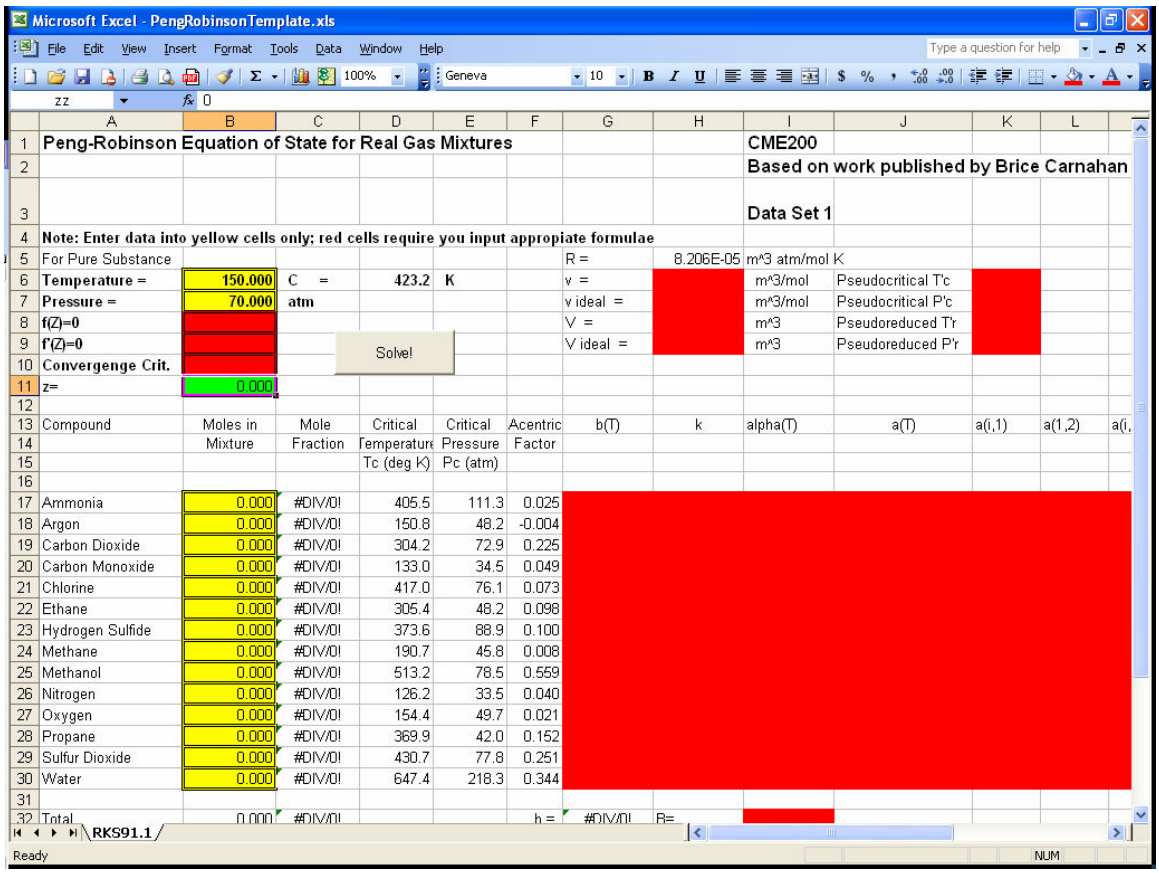

Figure 5. Spreadsheet used to calculate coefficients to populate the Peng-Robinson equation of state. Students were expected to program all cells highlighted in red. Yellow cells represent inputs to a simulation run. The green cell highlights the value to be calculated by the VBA subroutine. About 70% of the spreadsheet is visible.

Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition

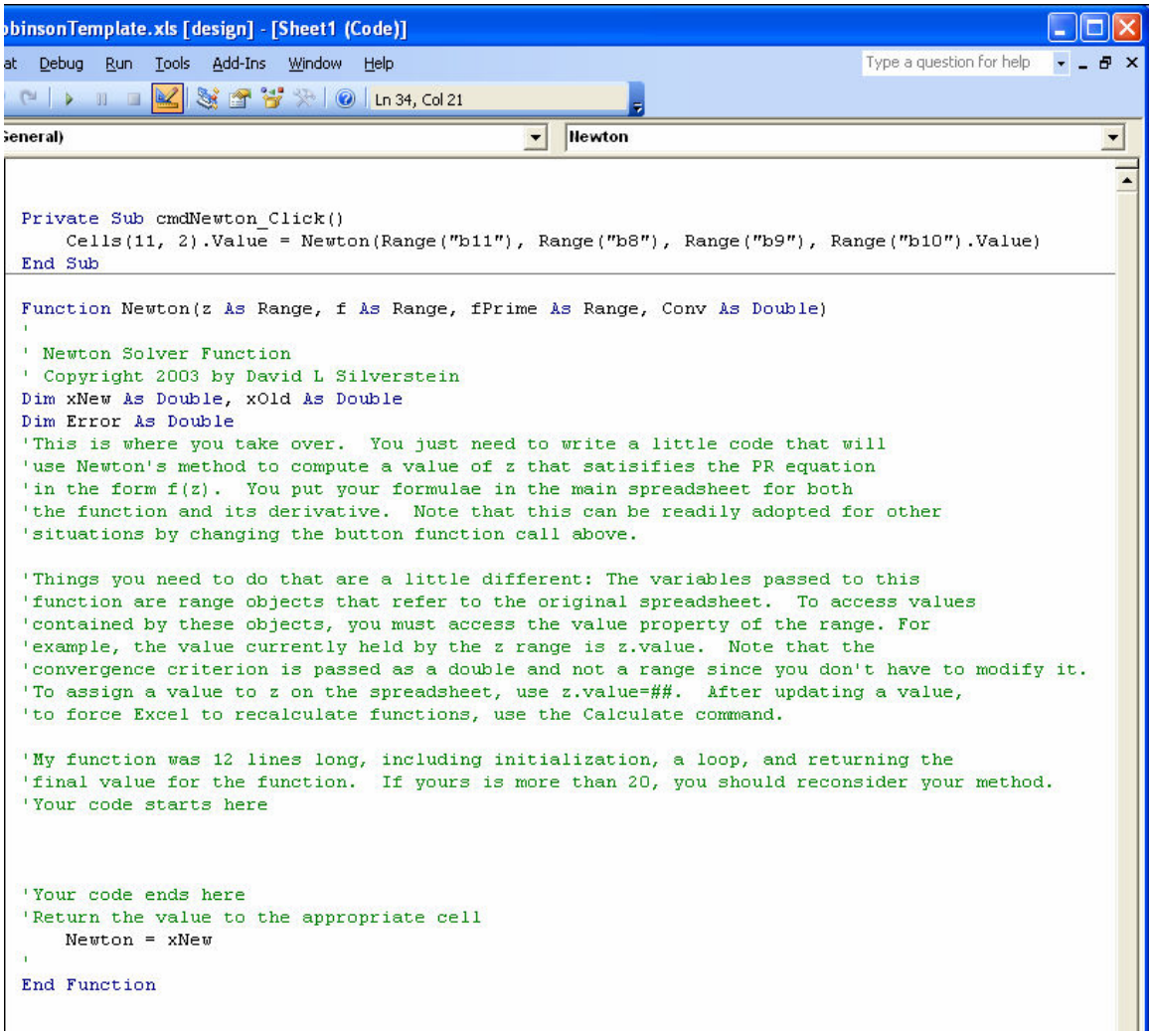

Figure 6. VBA module containing the subroutine called by the button on the spreadsheet and the function that it calls. The reason for the multiple levels of code was to isolate student code from the Excel generated response to the click event. A function could not be used alone in this case since the routine will need to evaluate updated values of the function to be solved and its derivative, which require that the spreadsheet be modified during the program execution. Other approaches would have reduced the general applicability of this Newton's method solver.